Proceedings of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 375–380
DOI: 10.15439/2025F9016
ISSN 2300-5963 ACSIS, Vol. 43

Metaheuristics for Rolling Stock Cyclic Job Scheduling Problems with Maintenance

Radosław Rudek
General Tadeusz Kościuszko Military University of Land Forces
Czajkowskiego 109, 51-147 Wrocław, Poland
Email:rudek.radoslaw@gmail.com

***Abstract*—In this paper, we analyse rolling stock cyclic job scheduling problems with maintenance to maximize the total availability of vehicles or to minimize the total maintenance cost. The considered issues can be formulated as cyclic job scheduling problems on parallel machines with time and usage based deteriorating effects and maintenance. Since these problems are strongly NP-hard, we propose a dedicated double crossover genetic algorithm and compare its efficiency with other metaheuristics: particle swarm optimization, simulated annealing, and genetic algorithm (using a single crossover). The computational experiments reveal that our approach is robust, optimizes various instances, and overwhelms the other evaluated algorithms for the analysed scenarios correlated with real-life cases. Thus, our new method is applicable for the industrial practice to maximized availability of vehicles as well as to minimize maintenance costs.**

***Index Terms*—Scheduling, Rolling stock, Deteriorating, Maintenance, Metaheuristic**

I. Introduction

SCHEDULING problems with deteriorating effects and/or maintenance activities have attracted particular attention in the research society (e.g., [1], [2], [3]). It follows from the practical meaning of the related research, since most of the industrial systems undergo degradation, and to avoid unexpected breakdowns preventive maintenance are performed. It is especially important in a railway domain, where the related rules concerning maintenance of vehicles are strictly defined. Usually, for locomotives five levels of activities are distinguished: inspections (Level 1), periodic inspections (Level 2), extended periodic inspections (Level 3), overhauls (Level 4), overhauls and upgrades (Level 5) [4]. However, they are not only time consuming and making vehicles unavailable [5], but they are also expensive (see [6]) especially for higher levels. On the other hand, the maintenance dates can be controlled within a certain range by scheduling taking into account required mileage of transport tasks and additional constraints following from maintenance policies defining time-based maintenance (TBM) and mileage based-maintenance (MBM) cycles. Such approach was for the first time analysed, developed and implemented by the author for a locomotive rental company, and his results including the dedicated mathematical (simulation) model and the optimization algorithms were described in details in the industrial report [7].

In this paper, we will follow that approach on rolling stock cyclic job scheduling problems with maintenance to maximize the total availability of vehicles or to minimize of the total maintenance cost. To solve them, we will propose a dedicated double crossover genetic algorithm and compare its efficiency with other metaheuristics known from the literature such as our previous implementations of particle swarm optimization [8], simulated annealing ([7], also presented in [5]) and a genetic algorithm with a single partially mapped crossover PMX [5] adjusted to solve the required criteria.

II. Problem formulation

In this section, we will describe the analysed rolling stock cyclic job scheduling problems to maximize the total availability of vehicles (TA) and to minimize the total maintenance cost (TC) as a cyclic job scheduling problems on identical parallel machines with deteriorating effects and maintenance. It is worth highlighting that these problems are covered by the mathematical (simulation) model, which was formulated solely by the author (R. Rudek) and precisely described in his industrial report [7]. Its fundamental part is the discrete event simulator (DES) including the model parameters and their relations describing vehicles, maintenance policies, jobs, schedules and rules allowing for determining dates and durations of maintenance activities and calculating various criteria. These main assumptions in a simplified form were presented to a broader audience *inter alia* in [9] and [5] to optimize the total availability criteria, the total maintenance cost [8] or the total income [10].

Let us recall that the discussed problems can be formulated as cyclic job scheduling problems on parallel machines with time and usage based deteriorating effects and maintenance under the these criteria. The scheduling problem (model) parameters and constraints are given as follows. There is a time horizon $\mathcal{T} = \{1, \ldots, t, \ldots, T\}$ is divided into T periods (in practice referring to succeeding months). A set $\mathcal{V} = \{1, \ldots, v, \ldots, n\}$ of n parallel machines (representing locomotives) that have to process a set $\mathcal{J} = \{1, \ldots, j, \ldots, n\}$ of n cyclic jobs (identified with transport tasks) during all periods T. A job can be assigned exactly to one and the same machine for a given period t, and a machine can process only one job at a time, but the reassignment in the next periods is allowed. Moreover each job $j \in \mathcal{J}$ is characterized by the average milage per day (workload) w_j that defines daily deteriorating of a machine (increasing its accumulative mileage) during all days in period t, when that machine is

operational (available) and processes job j. On the other hand machines, due to their deteriorating, must to undergo periodic preventive maintenance activities MAs, and each of them belongs to exactly one family (type) $f \in \mathcal{F} = \{1, \ldots, F\}$. Machines of a given type f share the same maintenance policy (MP), which defines time-based maintenance (TBM) and mileage (distance) based-maintenance (MBM) cycles divided into K^f levels of complexity. Each level $k = 1, \ldots, K^f$ of MP is described by the related parameters TBM_k^f and MBM_k^f, which are the predetermined threshold values triggering MA according to TBM or MBM, respectively. A maintenance activity of level k for a machine of type f is described by the expected duration MD_k^f and the cost MC_k^f. The maintenance activities of lower levels are covered by the higher levels and the following relations hold: $TBM_k^f < TBM_{k+1}^f$, $MBM_k^f < MBM_{k+1}^f$, $MD_k^f < MD_{k+1}^f$ and $MC_k^f < MC_{k+1}^f$ for all $k = 1, \ldots, K^f - 1$. Each machine $v \in \mathcal{V}$ is characterized by its deteriorating (condition) parameters $VT_{v,k}$ and $VM_{v,k}$ denoting the period and the accumulated mileage since the last maintenance of level k, respectively. The maintenance activity MA of level k for machine v of type f is started, when the first of the following deteriorating rules does not hold: $VT_{v,k} \leq TBM_k^f$ or $VM_{v,k} \leq MBM_k$ for all $k = 1, \ldots, K^f$ (referring to time and usage deteriorating). If a machine is out of order due to MA, then each job assigned to that machine during related unavailability periods is serviced by the third party. After MA of level k is completed (meaning also the completion of lower levers) the related deteriorating values (counters/conditions) are set to zero $VT_{v,i} = 0$ and $VM_{v,i} = 0$ for $i = 1, \ldots, k$ and a machine can process an assigned job (returned from the third party).

The considered problem is to find a schedule S that is an assignment of all jobs $j \in \mathcal{J}$ to all machines $v \in \mathcal{V}$ during all periods $t \in \mathcal{T}$, which optimizes the required criterion value Q, i.e., the total availability of machines (vehicles) TA or the total maintenance cost TC. In other words, the schedule $S = \langle S_1, \ldots, S_t, \ldots, S_T \rangle$ is the sequence of the assignments of jobs to machines (vehicles) $S_t = (S_t(1), \ldots, S_t(v), \ldots, S_t(n))$ in succeeding periods $t = 1, \ldots, T$, where $S_t(v) \in \mathcal{J}$ is the index of a job assigned to machine (vehicle) v at period t. Note that S can be represented unambiguously by the sequence of permutations S_t. Furthermore, a schedule determines the deteriorating of machines in particular periods (usually months) caused by processing of jobs, which together with model parameters and constraints (such as the maintenance policies) imply the maintenance plan (dates of maintenance activities) that is precisely described by our mathematical model RRR including the discrete event simulator [7]. Thus, the calculation of the criterion value $Q(S)$ for a given schedule S is based on the maintenance plan provided by our model, and it requires $O(MKn)$ steps, where M is the number of performed maintenance activities and $K = \max_{f \in \mathcal{F}}\{K^f\}$ is their maximum level for all machines. Therefore, the objective is to find a schedule S^* that optimizes the criterion value $Q(S)$, i.e., the maximization of the total availability of machines (vehicles) TA: $S^* = \arg\max_{S \in \mathbb{S}}\{Q(S)\} = \arg\max_{S \in \mathbb{S}}\{TA(S)\}$, the minimization of the total maintenance cost TC: $S^* = \arg\min_{S \in \mathbb{S}}\{Q(S)\} = \operatorname{argmin}_{S \in \mathbb{S}}\{TC(S)\}$, where $\mathbb{S}$ is the set of all feasible schedules.

III. Computational complexity

The rolling stock cyclic job scheduling problem to maximize the total availability of vehicles is strongly NP-hard [5]. Here, we will present a sketch of the strong NP-hardness proof of the analysed problem to minimize the total maintenance cost, since its full version is beyond the scope of this paper.

Theorem 1: The rolling stock cyclic job scheduling problem to minimize the total maintenance cost is strongly NP-hard.

Proof. The strong NP-hardness follows from the proof of the strong NP-hardness of the problem with the minimization of the total time on maintenance activities (see [5]). It can be done by extending the pseudopolynomial time transformation such that the parameter *maintenance cost* $MC = 1$ is added and the criterion $MA(S, T)$ is replaced by correlated the total cost of maintenance activities $CMA(S, T)$. Thus, it can be observed that $CMA(S, T)$ is equal to $y = 0$ only if $MA(S, T) = y = 0$ and $CMA(S, T) > y$ only if $MA(S, T) > y = 0$. □

Thus, it is highly unlikely (until $P = NP$) to construct an exact polynomial time algorithm for the considered problem. On the other hand, the solution space $\mathbb{S}$ is significantly greater than for typical scheduling problems, since it is spread by the sequence of permutations with the cardinality $|\mathbb{S}| = O((n!)^T)$. Therefore, we will proposed a dedicated double crossover genetic algorithm (GARDX) and compare its efficiency with other methods already published (e.g., [5], [8]).

IV. Genetic algorithm

In this section, we will present our genetic algorithm railway double crossover (GARDX), which is based on the general concept of the genetic algorithm metaheuristic [11]. In particular, it follows the construction for one permutation [12], later on extended to the sequence of permutations being our earlier basic implementation GA (see [5]) evaluated for the total availability criterion TA. Let us bring closer the idea behind GARDX, which is describe by Algorithm 1. It uses the mathematical model developed and implemented solely by the author (R. Rudek) [7] and included in his dll library `RudRobustRailway` (RRR). Since GARDX is designed to optimize arbitrary criteria Q that are based on the maintenance plan resulting from the schedule, the proper handler has to be chosen (e.g., TA or TC) and also the values of its meta parameters have to be given, i.e., $PopulationSize$, $OffspringSize$, $MixSize$, $StopConition$ (see step 1). Each individual in the population is expressed as the pair (S, Q), where $S = \langle S_1, \ldots, S_t, \ldots S_T \rangle$ is the schedule represented by the sequence of permutations S_t and Q is the corresponding criterion value (e.g., calculated according to TA or TC). Thus, we will refer to the schedule S and to the criterion value Q of the idxth element from a required set (e.g., $Population$) as follows: $Population[idx].S$ and $Population[idx].Q$ (step 2).

Algorithm 1 Genetic Algorithm Railway Double Crossover GARDX

```
 1: Input: parameters and criterion type Q(·);
 2: Refer to the schedule S and the criterion Q of
    the element idx from the set SetName by
    SetName[idx].S, SetName[idx].Q;
 3: Population[1] = provided by FA;
 4: Population[2] = provided by BH;
 5: Population[3] = provided by PSO (StopCondition=10s);
 6: Population[4] = provided by SA (StopCondition=10s);
 7: for p = 5,...,⌊PopulationSize/2⌋
 8:   S' = a sequence of random permutations;
 9:   Calculate the criterion Q' for S': Q' = Q(S');
10:   Population[p] = S' and Population[p].Q = Q';
11: end for
12: for p = ⌊PopulationSize/2⌋ + 1,...,PopulationSize
13:   S' = Population[1].S;
14:   for t = 1,...,T
15:     for r = 1,...,⌊n/2⌋
16:       u = random vehicle index from {1,...,n};
17:       v = random vehicle index from {1,...,n};
18:       Swap jobs between u and v in S'_t
19:     end for
20:   end for
21:   Calculate the criterion Q' for S': Q' = Q(S');
22:   Population[p] = S' and Population[p].Q = Q';
23: end for
24: while (StopCondition ≠ true)
25:   ParentPool = {choose 2*OffspringSize random
      elements from Population};
26:   for idx = 1,...,OffspringSize
27:     ParentXX = ParentPool[2*idx − 1];
28:     ParentXY = ParentPool[2*idx];
29:     t_a = random period index from {1,...,T};
30:     t_b = random period index from {1,...,T};
31:     (S^X, S^Y) = TPX(ParentXX.S, ParentXY.S, t_a, t_b);
32:     S^C = ROX(S^X, S^Y, MixSize);
33:     Calculate the criterion value Q^C
        for the child schedule S^C: Q^C=Q(S^C);
34:     Offspring[idx].S=S^C and Offspring[index].Q=Q^C;
35:   end for
36:   Population = Choose PopulationSize best elements
      from the set {Population ∪ Offspring}
      ordered from the best to the worst Q;
37:   if Population[1].Q is better than Q* then
38:     Q* = Population[1].Q and S* = Population[1].S;
39:   end if
40: end while
41: Return the schedule S* = (S*_1,...,S*_T)
    and its criterion value Q*;
```

Algorithm 2 Two Point Crossover (TPX): $(S^X, S^Y) = TPX(S^{XX}, S^{XY}, t_a, t_b)$

```
1: Input: parent schedules S^XX, S^XY and
   crossover parameters t_a, t_b;
2: if t_a > t_b then exchange values t_a and t_b;
3: for t = 1 to t_a − 1
4:   S^X_t = S^XX_t;
5:   S^Y_t = S^XY_t;
6: end for
7: for t = t_a to t_b
8:   S^X_t = S^XY_t;
9:   S^Y_t = S^XX_t;
10: end for
11: for t = t_b + 1 to T
12:   S^X_t = S^XX_t;
13:   S^Y_t = S^XY_t;
14: end for
15: Return resulting schedules S^X and S^Y;
```

Algorithm 3 Random Order Crossover (ROX): $S^C = ROX(S^X, S^Y, MixSize)$

```
 1: Input: parent schedules S^X, S^Y and
    parameter MixSize;
 2: Initialize the child schedule:
    S^C = ⟨S^C_1,...,S^C_t,...S^C_T⟩ = S^X;
 3: for q = 1,...,MixSize
 4:   t = random period index from {1,...,T};
 5:   for v = 1 to n
 6:     Get job index j assigned to vehicle v
        at period t for parent X: j = S^X_t(v);
 7:     Mark vehicle v in child schedule S^C_t as
        not assigned: S^C_t(v) = −1;
 8:     Mark job j as not assigned to the child:
        isJobAssigned[j] = false;
 9:     rv = random value from {0.0,...,0.9};
10:     if rv < 0.5 then
11:       Assign job j to vehicle v in S^C_t: S^C_t(i) = j;
12:       Mark j as assigned: isJobAssigned[j] = true;
13:     end if
14:   end for
15:   u = 1;
16:   for v = 1 to n
17:     Get job index j assigned to vehicle v
        at period t for parent Y: j = S^Y_t(v);
18:     if isJobAllocated[j] == false then
19:       while (S^C_t(u) > 0)
20:         u = u + 1;
21:       end while
22:       Assign job j to vehicle u in S^C_t: S^C_t(u) = j;
23:     end if
24:   end for
25: end for
26: Return the child schedule S^C = (S^C_1,...,S^C_T);
```

The first four individuals in the initial population (steps 3–6) are constructed by the following methods FA, BH, PSO and SA. Namely, Fixed Assignment (FA) is the constant assignment of jobs to vehicles during all months (no job reassignment between vehicles), i.e., a sequence of natural permutations (introduced as a default reference in [7], later on used in [5]). Balance Heuristic (BH) was developed and introduced in [7], later on used and shown to a broader audience in [9], whereas Particle Swarm Optimization (PSO) was presented in [8], and Simulated Annealing (SA) was introduced in [7], later on used in [5]. The applied PSO and SA optimize the criterion type chosen for GARDX and their meta parameter values are given in Section V, whereas their stop condition in GARDX is 10 seconds. Next $p = 5,\ldots,\lfloor PopulationSize/2 \rfloor$ individuals are the sequences of random permutations (steps 7-11). The rest of elements of the population $p=\lfloor PopulationSize/2 \rfloor+1,\ldots, PopulationSize$ are constructed on the basis of FA, where for each S_t jobs are swapped between two random vehicles u and v that are drawn $\lfloor n/2 \rfloor$ times (steps 12-23) for every $t=1,\ldots,T$.

In the each iteration of the main loop (steps 24-40), the algorithm chooses $OffspringSize$ random pairs from the population (step 25) that are the intermediate parents $ParentXX$ and $ParentXY$ (steps 27-28). Following them and for two random periods t_a and t_b (steps 29-30) the target parents are constructed represented by schedules S^X and S^Y (step 31). It is done by using the two point crossover TPX (Algorithm 2), which exchanges permutations of schedules

$ParentXX.S \equiv S^{XX}$ and $ParentXY.S \equiv S^{XY}$ in ranges $t \in \{t_a, \ldots, t_b\}$ (where $t_a < t_b$) returning the new schedules S^X and S^Y. On their basis, a new child schedule S^C is created following the random order crossover ROX and parameter $MixSize$ (step 32). The related procedure ROX described by Algorithm 3 initially clones the first parent schedule S^X to the child schedule S^C. Next, it draws $MixSize$ times random period indices $t \in \mathcal{T}$. For each t, the particular assignments of jobs to the vehicles from the permutation of the first parent S^X_t are copied with probability 0.5 to the related permutation of the child schedule S^C_t, whereas not chosen jobs are marked as unassigned. Subsequently, these missing jobs are assigned to unoccupied vehicles in S^C_t according to the order determined by the permutation of the second parent S^Y_t.

After the complete set $Offspring$ of $OffspringSize$ new schedules is generated, the new population is constructed. It is done by choosing $PopulationSize$ best solutions from the temporary set consisting the previous population and the new offspring, i.e., $\{Population \cup Offspring\}$ (step 36). If the new population contains a solution that is better than the best already found Q^*, then it is updated (steps 37-39). Note that *better* (step 37) for TA means that the new criterion value is greater than Q^*, whereas for TC means that it has to be smaller to be updated. Nevertheless, if the algorithm is implemented such that the availability TA is represented by positive and the cost TC by negative numbers, then we can unify the notation and focus only on the maximization, thereby the best found solution is updated only for greater values.

The computational complexity of a single iteration of GARDX (steps 24–40) is equal to $O(OffspringSize(MK+T)n)$ that is determined by TPX (step 31), ROX (step 32) and the calculation of $Q(\cdot)$ based on the maintenance plan (step 33), which are equal to $O(Tn)$, $O(Tn)$, $O(MKn)$, respectively.

V. Experiments

Let us now verify the accuracy of the proposed algorithm GARDX in reference to other methods implemented for similar problems. To provide reliable analysis of our algorithm, especially referring to the methods already described in the literature (particle swarm optimization PSO, simulated annealing SA, genetic algorithm GA), we will compare the scheduling algorithms for the instances constructed similarly as in the related works [5].[1] The set of all evaluated algorithms and their configurations are given as follows: Particle Swarm Optimization (PSO): $\omega = 0.6$, $c_1 = 0.5$, $c_2 = 0.5$, swarm size $m = 10$, bounds $X_{\min} = 0.0$, $X_{\max} = 4.0$, $\nu_{\min} = -4.0$, $\nu_{\max} = 4.0$ (following [8]); Simulated Annealing (SA): $Temp = 1000000$, $\alpha = 0.01$ (introduced in [7], later on used in [5]); Genetic Algorithm (GA): $PopulationSize = 200$, $OffspringSize = 50$, $MixSize = \lfloor 0.2T \rfloor$, (following [5]); Genetic Algorithm Railway Double Crossover (GARDX): $PopulationSize = 160$, $OffspringSize = 40$, $MixSize = \lfloor 0.2T \rfloor$. The initial solution (schedule) for each metaheuristic is determined by FA. To provide a fair comparison, due to nondeterministic nature of these methods, we run each of them 10 times and choose the best result (similarly as in [5]) as well as the stop condition for all metaheuristics is the same and set to 90s. Thus, we slightly extended the running times of PSO, SA and GA set in [8] and [5], giving them more time to traverse the search space.

[1]The algorithms were coded in C# (Microsoft Visual Studio 2022) and simulations were run under Windows 10 on PC, CPU Intel® Core™ i9-10885H 2.40GHz and 32GB RAM. All the corresponding software and algorithms, their previous as well as the current versions were coded solely by R. Rudek

TABLE I: Example preventive maintenance cycles, durations and costs of maintenance activities for Škoda 31E ($f = $ Š) and for E6ACTa Dragon 2 ($f = $ D) (see [10])

Type f	Level k	TBM^f_k	MBM^f_k [km]	MD^f_k [days]	MC^f_k [PLN*]
Š			Škoda 31E		
	1	14 days	2,500	0.125	1,000
	2	110 days	22,000	2	7,000
	3	32 months	220,000	21	50,000
	4	8 years	620,000	74	800,000
	5	16 years	1,200,000	84	1,000,000
D			E6ACTa Dragon 2		
	1	3 months	30,000	0.125	2,000
	2	1 year	150,000	2	12,000
	3	4 years	660,000	21	80,000
	4	8 years	1,220,000	74	1,000,000
	5	32 years	4,800,000	84	1,400,000

It is worth noticing that all the considered metaheuristics (PSO, SA, GA, GARDX) were developed and implemented solely by the author such that they can be applied for an arbitrary criterion based on his model developed in [7] and included in his dll library `RudRobustRailway` (RRR). Thereby, the only necessary action to optimize the required criterion by these algorithms is to choose and to apply a proper handler Q calculating the criterion value, e.g., referring to TA or TC; for Algorithm 1 see its step 33. Though our previous implementations SA, GA (e.g., [5]) and PSO [8] were originally used for different objective functions, they can be applied in the exactly same form to optimize also TA or TC (only by choosing a proper objective function handler).

Similarly as in [5] and [10], the computational experiments are provided for the instances constructed on the basis of two extreme types of vehicles representing one of the oldest (Škoda 31E series 181 produced till 1965) and one of the most modern locomotive (E6ACTa Dragon 2 produced since 2018) offered by rail vehicle rental companies in Europe. The maintenance cycles of these vehicles describing the time based TBM and the mileage based MBM maintenance policies, maintenance durations DM and costs MC are given in Table I. In the industrial practice, the maintenance activities of levels $k \geq 3$ start one day earlier than following from TBM^f_k [7]. Although the financial values are correlated with the market, they do not refer to any particular company. Moreover, if it is needed to analyse these values in US Dollars, the exchange rate to Polish Złoty (PLN) on 30 April 2025 was \$1 $\approx$ 3.77 PLN. The instances are based on $n \in \{10, 20\}$ vehicles of the same type

TABLE II: The total availability TA (in days), the improvement (additional days) δ_{TA} and the total loss (in days) Δ_{TA} to the best solution of the considered algorithms $A \in \{$FA, PSO, SA, GA, GARDX$\}$ for the analyzed instances

$T^{(y)}$ [years]	n	Measure [days]	FA	PSO	SA	GA	GARDX
Škoda 31E (series 181)							
5	10	TA	15,718	15,810	15,827	15,815	15,846
		δ_{TA}		92	109	97	128
		Δ_{TA}	128	36	19	31	0
	20	TA	31,448	31,546	31,568	31,585	31,624
		δ_{TA}		98	120	137	176
		Δ_{TA}	176	78	56	39	0
10	10	TA	31,060	31,268	31,330	31,356	31,367
		δ_{TA}		208	270	296	307
		Δ_{TA}	307	99	37	11	0
	20	TA	62,050	62,473	62,447	62,629	62,636
		δ_{TA}		423	397	579	586
		Δ_{TA}	586	163	189	7	0
15	10	TA	46,439	46,620	46,666	46,636	46,674
		δ_{TA}		181	227	197	235
		Δ_{TA}	235	54	8	38	0
	20	TA	92,877	93,026	93,168	93,061	93,183
		δ_{TA}		149	291	184	306
		Δ_{TA}	306	157	15	122	0
E6ACTa Dragon 2							
5	10	TA	17,820	17,911	17,914	17,914	17,915
		δ_{TA}		91	94	94	95
		Δ_{TA}	95	4	1	1	0
	20	TA	35,747	35,818	35,829	35,825	35,830
		δ_{TA}		71	82	78	83
		Δ_{TA}	83	12	1	5	0
10	10	TA	35,130	35,203	35,166	35,202	35,236
		δ_{TA}		73	36	72	106
		Δ_{TA}	106	33	70	34	0
	20	TA	70,360	70,407	70,370	70,381	70,458
		δ_{TA}		47	10	21	98
		Δ_{TA}	98	51	88	77	0
15	10	TA	52,519	52,602	52,555	52,519	52,625
		δ_{TA}		83	36	0	106
		Δ_{TA}	106	23	70	106	0
	20	TA	105,021	105,132	105,110	105,021	105,158
		δ_{TA}		111	89	0	137
		Δ_{TA}	137	26	48	137	0

TABLE III: The total maintenance cost TC, the financial savings δ_{TC} and the loss (in savings) Δ_{TC} to the best solution (all in thousand PLN, kPLN) of the considered algorithms $A \in \{$FA, PSO, SA, GA, GARDX$\}$ for the analyzed instances

$T^{(y)}$ [years]	n	Measure [kPLN]	FA	PSO	SA	GA	GARDX
Škoda 31E (series 181)							
5	10	TC	15,718	15,538	15,449	15,442	15,385
		δ_{TC}		180	269	276	333
		Δ_{TC}	333	153	64	57	0
	20	TC	31,906	31,518	31,243	31,311	31,200
		δ_{TC}		388	663	595	706
		Δ_{TC}	706	318	43	111	0
10	10	TC	36,482	33,563	36,132	33,269	33,181
		δ_{TC}		2,919	350	3,213	3,301
		Δ_{TC}	3,301	382	2,951	88	0
	20	TC	73,914	67,829	72,427	67,265	67,249
		δ_{TC}		6,085	1,487	6,649	6,665
		Δ_{TC}	6,665	580	5,178	16	0
15	10	TC	56,460	54,690	55,412	54,741	53,815
		δ_{TC}		1,770	1,048	1,719	2,645
		Δ_{TC}	2645	875	1,597	926	0
	20	TC	112,814	110,692	111,397	112,406	110,575
		δ_{TC}		2,122	1,417	408	2,239
		Δ_{TC}	2,239	117	822	1,831	0
E6ACTa Dragon 2							
5	10	TC	4,072	2,104	2,032	2,070	2,032
		δ_{TC}		1,968	2,040	2,002	2,040
		Δ_{TC}	2,040	72	0	38	0
	20	TC	6,152	4,210	4,074	4,150	4,074
		δ_{TC}		1,942	2,078	2,002	2,078
		Δ_{TC}	2,078	136	0	76	0
10	10	TC	15,784	13,660	13,730	13,664	13,604
		δ_{TC}		2,124	2,054	2,120	2,180
		Δ_{TC}	2,180	56	126	60	0
	20	TC	29,448	27,508	27,354	27,438	27,318
		δ_{TC}		1,940	2,094	2,010	2,130
		Δ_{TC}	2,130	190	36	120	0
15	10	TC	23,632	22,682	23,576	23,632	22,664
		δ_{TC}		950	56	0	968
		Δ_{TC}	968	18	912	968	0
	20	TC	47,424	46,354	47,114	47,424	46,272
		δ_{TC}		1,070	310	0	1,152
		Δ_{TC}	1,152	82	842	1,152	0

chosen from the considered locomotives. It is assumed that the simulations start at 01.01.2024 and at that day all vehicles are just after maintenance of the highest level K_f (overhaul) or they are just delivered from their producer, thereby $VT_{v,k}=0$ and $VD_{v,k}=0$ for $v=1,\ldots,n$ and $k=1,\ldots,K_f$ (identical parallel machines). The number of jobs is equal to the number of vehicles divided into $G=\lceil n/2 \rceil$ groups that share the same average mileage per day from range $w_j \in \{400,\ldots,700\}$ (see [5]). We consider periods referring to popular strategic planning time frame (from shorter to longer) $T^{(y)} \in \{5,10,15\}$ in years, i.e., $T \in \{60,120,180\}$ in months. The analysed algorithms $A \in \{$FA, PSO, SA, GA, GARDX$\}$ are evaluated for each instance I according to the following measures depending on the criterion $Q \in \{TA, TC\}$: the total availability of vehicles TA with $\delta_{TA}(A) \equiv \delta_{TA}(A,I) = TA(S^A) - TA(S^{FA})$ is the improvement (additional days) in reference to FA (higher values are better); $\Delta_{TA}(A) \equiv \Delta_{TA}(A,I) = TA^* - TA(S^A)$ is the total loss in days in reference to the best solution $TA^* = \max\{TA(S^A)\}$ for instance I (lower values are better); where $TA(S^A)$ is the total availability of vehicles (in days) for the schedule S^A provided by algorithm A; the total maintenance cost TC with $\delta_{TC}(A) \equiv \delta_{TC}(A,I) = TC(S^{FA}) - TC(S^A)$ is the improvement (financial savings) in reference to FA (higher values are better); $\Delta_{TC}(A) \equiv \Delta_{TC}(A,I) = TC(S^A) - TC^*$ is the total loss in savings in reference to the best solution $TC^* = \min\{TA(S^A)\}$ for instance I (lower values are better); where $TC(S^A)$ is the total maintenance cost for the schedule S^A provided by algorithm A. Note that Δ_{TA} and Δ_{TC} are redundant to δ_{TA} and δ_{TC}, respectively. Nevertheless, they are introduced for a more comprehensive comparison of the algorithms to clearly show differences between them and to easily recognize leading approaches (lower values are better).

The results of the computational experiments for instances based on the considered locomotive types, including criterion values TA (the total availability of vehicles in days) and TC (the total maintenance cost in thousand PLN), the improve-

ments to FA (δ_{TA}, δ_{TC}) and the losses to the best solutions (Δ_{TA}, Δ_{TC}) are shown in Tables II and III, respectively. It can be seen that all metaheuristics improved the results obtained by FA for both criteria. Our newly proposed GARDX overwhelms all other previous algorithms PSO, SA and GA for all considered instances under each of the separately optimized criteria: the total availability of vehicles TA and the total maintenance cost TC. It found schedules that offer additional measurable profits in reference to FA, e.g., from 128 additional days in 5 years and 10 locomotives to 586 days for 10 years and 20 locomotives (see Table II) and significant savings in the maintenance cost that exceed 2M PLN (c.a. \$0.5M) for $T^{(y)} = 5$ years and $n = 10$ locomotives or it can even reach 6M PLN (c.a. \$1.5M) for $T^{(y)} = 10$ and $n = 20$. The experiments revealed that GARDX is a robust method that well optimizes various instances, whereas other algorithms are instable. Therefore, GARDX can successfully replace each of the previous metaheuristics PSO, SA and GA for the optimization of TA or TA. Although the maximization of the total availability of vehicles TA is correlated with the minimization of the total maintenance cost TC, they are not equivalent. The optimization of TA does not need to lead to the identical improvement of TC, due to specifications and nonlinearity of these criteria. We have managed for Škoda instances to choose the costs of maintenance activities MC_k^f on particular levels (within the range of real values correlated with the market, see Table I) that allows us for the following analysis. Namely, for $T^{(y)} = 5$ and $n = 10$, FA obtained the maintenance plan that was characterized by TA equals to 15,718 days (Table II) and at the same time TC is equal to 15,718 thousand PLN (kPLN) (see Table III). Thus, we can see that the best result for TA is 15,846 days provided by GARDX and the ranking of algorithms is GARDX, SA, GA, PSO (see Table II), whereas the best value for TC is 15,385 kPLN obtained also by GARDX, but the ranking is different, i.e., GARDX, GA, SA, PSO (see Table III). GARDX optimizing TA for that instance was able to find the schedule S^{TA} resulting with a maintenance plan characterized by the best availability $TA(S^{TA}) = 15,846$ days (Table II) and the related total maintenance cost was $TC(S^{TA}) = 15,414$ kPLN. However, if GARDX optimized the criterion TC, then a different schedule S^{TC} was found corresponding to a maintenance plan characterized by the best total maintenance cost $TC(S^{TC}) = 15,385$ kPLN (Table III) and the related total availability in days was $TA(S^{TC}) = 15,826$. It can be seen that the total availability $TA(S^{TA}$ is better (greater) than $TA(S^{TC})$ if TA is optimized, and the total maintenance cost $TC(S^{TC})$ is better (smaller) than $TC(S^{TA})$ if the optimization objective is TC. Thereby, optimization of TA is not equivalent to optimization of TC.

VI. Conclusions

We analysed rolling stock cyclic job scheduling problems with maintenance to optimize the following objectives: the maximization of the total availability of vehicles TA and the minimization of the total maintenance cost TC. They were formulated as cyclic job scheduling problems on parallel machines with time and usage based deteriorating effects and maintenance. We constructed dedicated efficient genetic algorithm GARDX and we adjusted other metaheuristics PSO, SA, and GA known from the literature. The computational experiments revealed that GARDX overwhelms all compared methods and it proved its usefulness to the industrial practice to maximized TA and to minimize TC translating into convincing financial profits. For instance, it was able to significantly improve TA in reference to FA, e.g., from 128 additional days in 5 years and 10 locomotives to 586 days for 10 years and 20 locomotives. If it was applied to optimized TC, it found savings in reference to FA that exceeded 2M PLN (c.a. \$0.5M) for 5 years and 10 locomotives and 6M PLN (c.a. \$1.5M) for 10 years and 20 vehicles.

Acknowledgement

This work was supported by the Polish National Science Centre under grant no. DEC-2020/37/B/HS4/03235 and by General Kościuszko Military University of Land Forces (presentation and computational experiments). We would like to thank Anna Litwa-Janeczek and Tomasz Marchel for analysis concerning criteria and intermediate simulations of algorithms.

References

[1] M. Atsmony, B. Mor, and G. Mosheiov, "Minimizing tardiness scheduling measures with generalized due-dates and a maintenance activity," *Computers & Operations Research*, vol. 152, p. 106133, 2023.

[2] R. P. Nicolai and R. Dekker, "Optimal maintenance of multi-component systems: a review," in *Complex system maintenance handbook*. London: Springer Science & Business Media, 2008, pp. 263–286.

[3] X. Sun and X.-N. Geng, "Single-machine scheduling with deteriorating effects and machine maintenance," *International Journal of Production Research*, vol. 57, pp. 3186–3199, 2019.

[4] C. Zhang, Y. Gao, L. Yang, Z. Gao, and J. Qi, "Joint optimisation of train scheduling and maintenance planning in a railway network: A heuristic algorithm using Lagrangian relaxation," *Transportation Research Part B: Methodological*, vol. 134, pp. 64–92, 2020.

[5] R. Rudek and I. Rudek, "Models and algorithms for the preventive maintenance optimization of railway vehicles," *Expert Systems with Applications*, vol. 240, pp. 122 589.1–13, 2024.

[6] E. Eisenschmidt, S. Reimig, L. Schirmers, and S. Stern, "The rail sector's changing maintenance game," 2017, mcKinsey Global Institute, Report, https://tinyurl.com/ymbup3v5 [accessed: 2024-08-20].

[7] R. Rudek, "A decision support system for rolling stock management in a railway transport company," 2018, Industrial Report "Mozart Project" for Industrial Division Ltd., October 2017 – September 2018 (in Polish).

[8] I. Rudek and R. Rudek, "Optimizing maintenance cost of uniform rolling stock by scheduling algorithms," in *Lecture Notes in Networks and Systems*, L. Borzemski, H. Selvaraj, and J. Swiatek, Eds. Switzerland: Springer Nature, 2022, vol. 364, pp. 261–270.

[9] I. Heppner and R. Rudek, "Improve railway vehicle availability by scheduling under preventive maintenance policies," in *48th International Conference on Computers & Industrial Engineering 2018 (CIE48)*, X. Xun, M. I. Dessouky, and R. Y. Zhong, Eds. Computers and Industrial Engineering, 2018, pp. 2325–2332.

[10] R. Rudek, "Local search algorithms for a railway scheduling problem under maintenance and cost criteria," in *Models and Methods for Systems Engineering. Studies in Big Data*, G. Borowik, G. Chmaj, and R. Waszkowski, Eds. Cham: Springer, 2025, vol. 165, pp. 39–49, https://doi.org/10.1007/978-3-031-76440-0_4.

[11] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer-Verlag, 1996.

[12] R. Rudek, "A generic optimization framework for scheduling problems under machine deterioration and maintenance activities," *Computers & Industrial Engineering*, vol. 174, pp. 108 800.1–22, 2022.